CERN -TH/98-127

MPI-PhT/98-34

Ordering monomial factors of polynomials in the product representation

B. Bunk[1], S. Elser[1], R. Frezzotti[2] and K. Jansen[3]

[1] Institut für Physik, Humboldt–Universität, Invalidenstr. 110, 10115 Berlin, Germany

[2] Max-Planck-Institut für Physik, Föhringer Ring 6, D-80805 München, Germany

[3] CERN, 1211 Genève 23, Switzerland

February 1, 2008

Abstract

The numerical construction of polynomials in the product representation (as used for instance in variants of the multiboson technique) can become problematic if rounding errors induce an imprecise or even unstable evaluation of the polynomial. We give criteria to quantify the effects of these rounding errors on the computation of polynomials approximating the function $1/s$. We consider polynomials both in a real variable s and in a Hermitian matrix. By investigating several ordering schemes for the monomials of these polynomials, we finally demonstrate that there exist orderings of the monomials that keep rounding errors at a tolerable level.

1 Introduction

In Monte Carlo simulations of fermionic systems in a discretized space-time, the determinant of a matrix A, related to the lattice action of the fermions, usually has to be taken into account. Although the form of the matrix A can be very general and depends on the problem under consideration, we assume in the following that A defines a *local* action of the fermions extending only over a few lattice spacings. A standard way to incorporate $\det A$ into simulation algorithms is to write it as a Gaussian integral

$$\det A = \int \mathcal{D}\Phi^\dagger \mathcal{D}\Phi e^{-\Phi^\dagger A^{-1}\Phi} \tag{1}$$

where Φ is a suitable complex N-component vector on which the matrix A acts and $\mathcal{D}\Phi^\dagger\mathcal{D}\Phi$ the corresponding (properly normalized) integration measure. One is therefore led to the inconvenient problem of inverting an $N \otimes N$ matrix, which can be very large, i.e. N being $\mathrm{O}(10^6)$.

In [1] an alternative approach was introduced: the determinant of A may be approximated by the inverse determinant of a polynomial $P_n(A)$ of degree n in the matrix A such that

$$\det A \approx \left[\det P_n(A)\right]^{-1} \ . \tag{2}$$

In eq. (2) and throughout the rest of the paper we assume that A is Hermitian, positive definite, and that $\|A\| \leq 1$, where $\|A\|$ is given by the largest eigenvalue $\lambda_{\max}(A)$ of A. In the following, we will consider only polynomials of even degree n. The roots z_k, $k = 1, ..., n$, of the polynomial hence come in complex-conjugate pairs and the determinant $\det P_n$ can be factorized into positive factors, resulting in

$$\left[\det P_n(A)\right]^{-1} \propto \prod_{k=1}^{n/2} \left[|\det(A - z_k)|^2\right]^{-1} \propto \prod_{k=1}^{n/2} \int \mathcal{D}\Phi_k^\dagger \mathcal{D}\Phi_k e^{-\sum_k \Phi_k^\dagger (A-z_k)^\dagger (A-z_k)\Phi_k} \ . \tag{3}$$

From eq. (3), we can see that now the action of the bosonic fields Φ_k is local and hence the task of inverting the matrix A can be avoided. Similar steps lead to the so-called multiboson technique for simulating fermionic systems. This technique has been shown to give a comparable performance [2–8] to the standard Hybrid Monte Carlo (HMC) [9] or Kramers equation [10, 11] simulation algorithms. Examples for applications of the multiboson technique are Monte Carlo simulations of lattice QCD [3, 6], supersymmetry [12], the Schwinger model [8] and the Hubbard model [13].

In exact versions of the multiboson technique [4] or related approaches [14, 15, 16] the use of a product representation of the polynomial $P_n(A)$ often turns out to be convenient. However, the numerical construction of a polynomial using the product representation can –because of rounding errors– easily lead to a loss of precision or even to numerical instabilities. This holds true in particular if computers with 32-bit arithmetic precision are used. Motivated by simulation algorithm studies, which use the product representation of a polynomial, we will show in this paper that by suitable orderings of the monomial factors, the precision losses can be kept on a tolerable level. Although we will only demonstrate this for a particular example of the matrix A, we expect that similar results may also hold for more general situations.

The paper is organized as follows: in section 2 we will specify the polynomial we have used in this work. Effects and possible origins of rounding errors when the polynomial is evaluated in its product representation are discussed. In section 3 we give the ordering schemes for the monomials of the polynomial in the product representation. We show how the evaluation of the polynomial is affected when the different ordering schemes are employed. Section 4 is devoted to quantitative estimates of the rounding-error effects. We compare in section 5 some results when using 32-bit and 64-bit arithmetics and conclude in section 6.

2 Product representations of polynomials and rounding errors

Let us consider the approximation of a function $f(s)$ depending on a real variable $s > 0$ by a polynomial $P_n(s)$ of degree n. The motivation to initially study a single degree of freedom, is –besides its simplicity– that we might think of the matrix A as being diagonalized. Then the problem, eq. (2), reduces to finding a polynomial that approximates each $\lambda^{-1}(A)$ separately, where $\lambda(A)$ is a real eigenvalue of A. We therefore expect that studying a single degree of freedom can provide information also about the qualitative behaviour of rounding-error effects when the polynomial $P_n(A)$ in the matrix A is computed.

To be specific, we follow the Chebyshev approximation method [17, 20] and construct a polynomial approximating the function $f(s) = 1/s$ in the range $\epsilon \leq s \leq 1$, where $\epsilon \geq 0$ is an adjustable parameter. The choice of the function $1/s$ is, of course, motivated by our original problem of evaluating an inverse matrix, eq. (1). We take the same polynomial $P_{n,\epsilon}(s)$ as specified in [18]. For completeness, we recall here its definition. If we introduce the scaled variables u and θ

$$u = (s-\epsilon)/(1-\epsilon)\ ,\ \ \cos\theta = 2u - 1 \tag{4}$$

the Chebyshev polynomial $T_r^*(u)$ of degree r is given by (note that T_r^* is *not* the standard definition of the Chebyshev polynomial):

$$T_r^*(u) = \cos(r\theta)\ . \tag{5}$$

For given n and ϵ the polynomial $P_{n,\epsilon}(s)$ is then defined by

$$P_{n,\epsilon}(s) = \left[1 + \rho T_{n+1}^*(u)\right]/s\ , \tag{6}$$

where the constant ρ has to be taken such that the square bracket vanishes at $s = 0$. The polynomial eq. (6) approximates the function $1/s$ uniformly in the interval $\epsilon \leq s \leq 1$. The relative fit error

$$R_{n,\epsilon}(s) = \left[P_{n,\epsilon}(s) - 1/s\right] s \tag{7}$$

in this interval decreases exponentially with increasing n

$$|R_{n,\epsilon}(s)| \leq \delta \equiv 2\left(\frac{1-\sqrt{\epsilon}}{1+\sqrt{\epsilon}}\right)^{n+1}\ . \tag{8}$$

The accuracy parameter δ provides an upper bound for the absolute value of the relative fit error in the given interval. We note in passing that $|\rho| \leq \delta$ and that in all the practical applications studied in this paper the value of ρ is actually very close to the value of δ.

The roots z_k of the polynomial eq. (6) can be computed analytically,

$$z_k = \frac{1}{2}(1+\epsilon) - \frac{1}{2}(1+\epsilon)\cos\left(\frac{2\pi k}{n+1}\right) - i\sqrt{\epsilon}\sin\left(\frac{2\pi k}{n+1}\right)\ . \tag{9}$$

We then obtain the desired product representation of the polynomial

$$P_{n,\epsilon}(s) = \prod_{k=1}^{n}\left[c_k(s - z_k)\right]\ . \tag{10}$$

The real normalization factors c_k have to satisfy the condition

$$\prod_{k=1}^{n} c_k = C_{\mathrm{tot}}(n) = \left(\frac{1+\epsilon}{2}\prod_{k=1}^{n}\left[\frac{1+\epsilon}{2} - z_k\right]\right)^{-1} . \tag{11}$$

When the c_k's are taken to be all identical, they turn out to be $O(1)$. We will also use later the partial products

$$P_{n,\epsilon}^{l}(s) = \prod_{k=1}^{l}\left[c_k(s - z_k)\right] . \tag{12}$$

One may also be interested [16] in a product representation of the polynomial $P_{n,\epsilon}(s) = p_{n,\epsilon}(q)$ in terms of the real variable $q = \pm\sqrt{s}$:

$$p_{n,\epsilon}(q) = \prod_{k=1}^{2n}\left[\sqrt{c_k}(q - r_k)\right] , \tag{13}$$

where the roots r_k are defined as

$$\begin{aligned} r_k &= \sqrt{z_k},\ \mathrm{Im}(\sqrt{z_k}) > 0\ ,\ k = 1, \ldots, n \\ r_k &= r_{2n+1-k}^{*}\ , \quad k = n+1, \ldots, 2n \end{aligned} \tag{14}$$

with the obvious generalization for the constants $\sqrt{c_k}$. The representation of eq.(13) is used in the algorithm described in [15], where it leads to a most efficient implementation of the so-called PHMC algorithm [16].

The above definition of the polynomial $p_{n,\epsilon}(q)$, and hence also $P_{n,\epsilon}(s)$, can straightforwardly be generalized to the case when, instead of the real variable q, a Hermitian matrix is used. In this paper we will only study the special matrix $\hat{Q}$ and refer to appendix A for its definition. Here we only mention that $\hat{Q}$ has a band structure with only the diagonal and a few off-diagonals different from zero. We will use for our studies of rounding-error effects both forms of the polynomial, eq.(10) and eq.(13). We already remark at this point that no qualitative difference in our results could be seen using either of the two forms of the polynomial.

In order to discuss the rounding errors occurring in the evaluation of the polynomial $p_{n,\epsilon}$ either in a real variable or in a matrix, let us consider the quantities

$$\begin{aligned} \omega_l(q) &\equiv |\sqrt{c_l}(q - r_l)\ldots\sqrt{c_1}(q - r_1)q|, \\ \Omega_l(\hat{Q}) &\equiv \|\sqrt{c_l}(\hat{Q} - r_l)\ldots\sqrt{c_1}(\hat{Q} - r_1)\hat{Q}\Phi_{\mathrm{in}}\|/\|\Phi_{\mathrm{in}}\|, \\ & \quad\ l \in \{1, 2, ..., 2n+1\}\ , \quad r_{2n+1} = 0, \quad c_{2n+1} = 1\ . \end{aligned} \tag{15}$$

In eqs.(15), $\Phi_{\rm in}$ is a suitable vector on which the matrices act. The constants r_{2n+1} and c_{2n+1} are chosen such that the last factor in $\omega_{2n+1}(q)$ and $\Omega_{2n+1}(\hat{Q})$ corresponds to a multiplication by q and $\hat{Q}$, respectively.

In practice the values of subsequent products, such as $\omega_l(q)$ and $\omega_{l+1}(q)$, may differ substantially. This can be clearly seen in Fig. 1a. For the figure we have chosen $n = 64$ and $\epsilon = 0.0015$, leading to a relative fit accuracy of $\delta \simeq 0.013$. The values of n and ϵ are motivated by the values used in practical applications such as simulations of lattice QCD.

We plot $\omega_l(q)$ for values of q at the low end of the interval, $q^2 = \epsilon$, the middle of the interval, $q^2 = (1+\epsilon)/2$, and the upper end of the interval, $q^2 = 1$. We will restrict ourselves to positive values of q. Using the roots as given in eq.(14) and taking the factors $\sqrt{c_k}$ all identical, this restriction induces no loss of generality because of the relation $\omega_l(q)\omega_n(-q) = q\omega_{n+l}(-q)$, $l \in [1, n]$ and $q \neq 0$. In the case considered in Fig. 1a the oscillations of $\omega_l(q)$ do not lead to numerical overflows. Consequently, the final value $\omega_{2n+1}(q)$ comes out to be close to 1, with a deviation consistent with the value of the accuracy parameter δ.

The observed behaviour of $\omega_l(q)$ leads us to expect that large precision losses may affect the evaluation of the same polynomial in the matrix $\hat{Q}$, when using a product representation, as we do for the computation of $\Omega_l(\hat{Q})$, eq.(15). The l-th step of this computation, yielding Φ_l as a result, amounts to the multiplication of the vector

$$\Phi_{l-1} = \sqrt{c_{l-1}}(\hat{Q} - r_{l-1}) \cdot \ldots \cdot \sqrt{c_1}(\hat{Q} - r_1)\hat{Q}\Phi_{\rm in} \qquad (16)$$

by the matrix $\sqrt{c_l}(\hat{Q} - r_l)$. In order to understand the relation between the quantities $\omega_j(q)$ and the rounding errors on $\Omega_j(\hat{Q}) = \|\Phi_j\|/\|\Phi_{\rm in}\|$, it is useful to think of the vectors Φ_j and $\Phi_{\rm in}$ as linear combinations of the eigenvectors of $\hat{Q}$:

$$\Phi_j = \sum_b \langle q_b|\Phi_{\rm in}\rangle \sqrt{c_j}(q_b - r_j) \cdot \ldots \cdot \sqrt{c_1}(q_b - r_1)|q_b\rangle \ , \qquad (j = 1, 2, \ldots, 2n+1) \ , \quad (17)$$

where $\hat{Q}|q_b\rangle = q_b|q_b\rangle$ and $\Phi_{\rm in}$ is assumed to have projections generally non vanishing on all the eigenvectors of $\hat{Q}$. Let us now go back to situations like the one in Fig. 1a where, for several integer values of l, the quantities $\omega_l(q)$ in a given range of values of q turn out to be much larger than for all other values of q and they also change substantially as a function of l. It is clear that in such situations the quantities $\Omega_l(\hat{Q})$ must substantially change with l, too. In particular the situation $\Omega_l(\hat{Q}) \ll \Omega_{l-1}(\hat{Q})$ must occur for some values of l, if the correct final value $\Omega_{2n+1}(\hat{Q}) \simeq \|\Phi_{\rm in}\|$ is going to be obtained. But such a situation can only be the result of substantial cancellations in the multiplication leading from Φ_{l-1} to Φ_l: it is here that we expect the occurrence of large rounding errors, which then propagate through the whole computation. As we will see later, these anticipated precision losses actually occur when the polynomial in a matrix is evaluated through its product representation.

The problem itself suggests, however, its solution: the monomial factors in eq.(10) or eq.(13) should be ordered, if possible, in such a way that the absolute values of all subsequent products of monomials in eq.(10) or eq.(13) have the same order of magnitude. Of course, whenever possible, evaluating a polynomial in the product representation should be avoided, because in general numerically stable recursion relations [17] or other numerical recipes [20] are available. However, for some cases, as discussed in the introduction, one has to rely on the product representation of the polynomial.

3 Ordering schemes

In this section we want to introduce the different ordering schemes, that we use for the monomials in eqs.(10) and (13), and the well-known, numerically stable Chebyshev method for the evaluation of general polynomials. Throughout this paper we will use the homogeneous distribution

$$c_k = (C_{\rm tot})^{\frac{1}{n}}$$

for the normalization constants. We remark that in principle one could try, at least for the case when a polynomial in a matrix is considered, to also distribute the normalization constants c_k in a k-dependent way to reduce rounding errors. However, as expected, we only found a very weak dependence of the rounding errors in the matrix case on the distribution of the c_k's.

3.1 Definition of ordering schemes

We start by defining ordering schemes for the monomial factors in eq.(10), or equivalently the roots z_k of eq.(9).

Naive ordering

As naive we regard the ordering given by

$$z_k^{\mathrm{naive}} = z_k \ , \qquad k = 1, \cdots, n \ , \tag{18}$$

where the roots z_k, given in eq. (9), lie on an ellipse in the complex plane. In the naive ordering the roots are selected from this ellipse by starting at the origin and moving anti-clockwise. This is indicated in Fig. 2a, where the roots are shown labelled according to the order in which they are used in the evaluation of the polynomial of eq. (10). Adopting this ordering of the roots for the construction of the polynomial in a matrix according to its product representation gives rise to substantial rounding-error effects.

Pairing scheme

A first improvement over the naive ordering is to use a simple pairing scheme, which amounts to reordering the roots as follows:

$$z_k^{\mathrm{pair}} = z_{j(k)}^{\mathrm{naive}} \ , \qquad k = 1, \ldots, n \ .$$

Let us give the reordering index $j(k)$ for the example of n being a multiple of 8 and $n' = n/8$. In the lower half-plane, Im $z_k < 0$, the pairing scheme is achieved by

$$\begin{aligned} j \ = \ & \Big\{ 1, \frac{n}{2}, \frac{n}{4}+1, \frac{n}{4}, \\ & 2, \frac{n}{2}-1, \frac{n}{4}+2, \frac{n}{4}-1, \\ & \ldots \\ & n', \frac{n}{2}-n'+1, \frac{n}{4}+n', \frac{n}{4}-n'+1 \Big\} \end{aligned} \tag{19}$$

and for Im $z_k > 0$ correspondingly. An illustration of the ordering in the pairing scheme is shown in Fig. 2b.

In the case where $n/2$ is not divisible by 4, we search for the next integer m, which is smaller than $n/2$ and divisible by 4. We then repeat the above described procedure on these m roots and simply multiply the remaining roots $z_{m+1} \cdots z_{n/2}$ at the end.

Subpolynomial scheme

The problem of precision losses in evaluating a polynomial in a matrix according to its product representation becomes more severe in general when increasing the degree n and in the specific case of $P_{n,\epsilon}(s)$, eq. (6), also when decreasing ϵ. In order to reduce the effects of rounding errors, one may therefore be guided by the following intuition. Let us consider the polynomial $P_{n,\epsilon}(s)$ with roots z_k and $n \gg 1$. If m is an integer divisor of n, the roots $z_1, z_{1+m}, z_{1+2m}, \ldots, z_{1+(n/m-1)m}$ turn out to be close to the roots characterising the polynomial $P_{n',\epsilon}(s)$ of degree $n' = n/m$ (note that we keep the same ϵ). Moreover, the normalization constants $c_k = (C_{\text{tot}}(n))^{\frac{1}{n}}$ and $c'_k = (C_{\text{tot}}(n'))^{\frac{1}{n'}}$ are of the same order (the dependence on n of c_k turns out to be negligible for large n). Then the product

$$u(s) = \prod_{j=0}^{n/m-1} \left[c_{j+1}(s - z_{1+jm})\right] \qquad (20)$$

is a rough approximation of $P_{n',\epsilon}(s)$, $|u(s) - P_{n',\epsilon}(s)| < 1$ for all $\epsilon \leq s \leq 1$. The same argument may be repeated for the other similar sequences of roots, like $z_2, z_{2+m}, z_{2+2m}, \ldots z_{2+(n/m-1)m}, \ \ldots, \ z_m, z_{2m}, z_{3m}, \ldots z_n$.

This means that the product eq.(10) may be split into a product of m subpolynomials, in such a way that each of them roughly approximates a polynomial $P_{n',\epsilon}(s)$ of *lower* degree $n' = n/m$. Because of the lower degree of the subpolynomials given by products such as eq. (20), one may expect that only small changes in the magnitude of the partial products occur in the intermediate steps of the evaluation of each of these subpolynomials.

The reordering of the subpolynomial scheme

$$z_k^{\text{sp}} = z_{j(k)}^{\text{naive}} \ , \qquad k = 1, \ldots, n$$

can be represented by

$$\begin{aligned} j \ = \ & \Big\{1, 1+m, 2+2m, \ldots, 1 + \left(\frac{n}{m} - 1\right) m, \\ & 2, 2+m, 2+2m, \ldots, 2 + \left(\frac{n}{m} - 1\right) m, \\ & \ldots \\ & m, m+m, m+2m, \ldots, m + \left(\frac{n}{m} - 1\right) m\Big\} \ , \end{aligned} \qquad (21)$$

where m is an integer divisor of n.
We found that m has to be chosen as $m \approx \sqrt{n}$ in order to minimize the changes in the magnitude of the partial products occurring in the intermediate steps of the construction of $P_{n,\epsilon}(s)$. We remark that the naive ordering is reproduced by the two extreme choices $m = 1$ and $m = n$.

Bit-reversal scheme
The subpolynomial scheme can be generalized, leading to what we will call the bit-reversal scheme. To illustrate how this scheme works, let us assume that the degree n of the polynomial is a power of 2. One now starts with the n monomial factors in eq. (10), chooses $m = n/2$ and applies the subpolynomial scheme resulting in m binomial factors. We then proceed to choose an $m' = m/2$ and again applying the subpolynomial scheme to these m binomial factors which leaves us with m' subpolynomials each of degree 4. The procedure can be iterated until we are left with only one subpolynomial having the degree of the polynomial itself.
The above sketched procedure can be realized in practice by first representing the integer label (counting from 0 to $n-1$) of the roots in the naive order by its bit representation. The desired order is then obtained by simply reversing the bits in this representation. The resulting reordering of the roots is shown in Fig. 2c, with $n = 16$ as an example.
For n not a power of 2, we pad with dummy roots, chosen to be zero for instance, until the artificial number of roots is a power of 2. The bit-reversal procedure can then be applied as described above. Afterwards, the dummy roots have to be eliminated from the sequence.

Montvay's scheme
Recently, Montvay [19] suggested to order the roots according to an optimization procedure that can be implemented numerically. Let us shortly sketch how Montvay's ordering scheme works and refer to [19] for further details. Let us assume that we have already the optimized order of the roots for the partial product $P^l_{n,\epsilon}(s)$, eq. (12). Then the values of $|sP^l_{n,\epsilon}(s)(s-z)|$ are computed for all z taken from the set of roots not already used. The values of s are taken from a large enough discrete set of points, $\{s_1, \ldots, s_N\}$, which are all in the interval $[\epsilon, 1]$.

Now, the maximal ratio over $s \in \{s_1, \ldots, s_N\}$ of all values $|sP^l_{n,\epsilon}(s)(s-z)|$, i.e.

$$\max_{s \in \{s_1,\ldots,s_N\}} |sP^l_{n,\epsilon}(s)(s-z)| / \min_{s \in \{s_1,\ldots,s_N\}} |sP^l_{n,\epsilon}(s)(s-z)| \ ,$$

is computed for each root z separately. Finally that root is taken which gives the *lowest* of these maximal ratios. Starting with the trivial polynomial $P^0_{n,\epsilon}(s) = 1$, this procedure obviously defines a scheme according to which the roots can be ordered iteratively. We show in Fig. 2d the resulting order of the roots using Montvay's scheme by again labelling the roots in the order in which they are used to compute the polynomial of eq. (10). It is clear that Montvay's ordering scheme implements by construction our intuitive criterion that the changes in the magnitude of subsequent partial products should be minimized.

Ordering schemes for the roots $\{r_j\}$

The above discussion concerned the ordering of monomial factors in eq.(10). If one wants to use the product representation, eq.(13), the ordering of the monomial factors can be obtained by first ordering the roots z_k in one of the ways described above and then defining the $2n$ roots r_j as in eq.(14). The one exception is the case of Montvay's scheme, where the r_j's themselves have to be ordered, in full analogy with the procedure desribed above for the case of the product representation (10). We remark that the ordering schemes for $\{r_j \ : \quad j = 1, \ldots, 2n\}$ which we consider in the present paper, always satisfy the relation in the second line of eq.(14). Although there is in principle much more freedom, this restriction turns out to be highly convenient for the application of Monte Carlo simulation algorithms.

3.2 Clenshaw recursion

Any polynomial $\mathcal{P}_m(u)$ of degree m in the real variable u, with $u \in [0,1]$, can be expressed as a linear combination of Chebyshev polynomials $T^*_k(u)$

$$\mathcal{P}_m(u) = \frac{1}{2} a_0 T^*_0(u) + \sum_{k=1}^{m} a_k T^*_k(u)$$

where $a_0, a_1 \ldots, a_m$ are suitable coefficients and the definition of $T^*_k(u)$ is given by eq.(5) and $\cos\theta = 2u - 1$.

A way [17, 20] of computing $\mathcal{P}_m$ is to use the formula

$$\mathcal{P}_m(u) = \frac{1}{2}(b_0(u) - b_2(u)) \ , \qquad (22)$$

where the right-hand side has to be evaluated through the recursion relation:

$$\begin{aligned} b_{m+2}(u) &= b_{m+1}(u) = 0 \\ b_k(u) &= 2(2u-1)b_{k+1}(u) - b_{k+2}(u) + a_k \ , \end{aligned} \qquad (23)$$

for all integers k starting from m down to 0. It is also possible to prove that the total rounding error on the final result $(b_0(u) - b_2(u))/2$ cannot exceed the arithmetic sum of the rounding errors occurring in each step of (23).

This well-known, numerically stable method [17],[20] can be applied to the evaluation of $P_{n,\epsilon}(s)$ as well as of $sP_{n,\epsilon}(s)$. In the latter case, if we consider $sP_{n,\epsilon}(s)$ as a polynomial of degree $n+1$ in $u = (s-\epsilon)(1-\epsilon)$, its expression as a linear combination of Chebyshev polynomials in u can be read from eq.(6):

$$sP_{n,\epsilon}(s) = T_0^*(u) + \rho T_{n+1}^*(u) \ .$$

We can then evaluate $sP_{n,\epsilon}(s)$ through the Clenshaw relation (23) by taking $a_0 = 2$, $a_{n+1} = \rho$ and all the other a_k's vanishing. The Clenshaw recursion will serve us in the following as a reference procedure for the numerical evaluation of the polynomial $\hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2)$.

3.3 Ordering schemes at work: a first look

As discussed in section 2, the large oscillations of $\omega_l(q)$, eq. (15), can easily lead to precision losses when constructing the same polynomial in a matrix. This can be seen in Fig. 1b for the case of the matrix $\hat{Q}$ (see Appendix A for the definition of the matrix $\hat{Q}$). There we have considered an $8^3 \cdot 16$ lattice with gauge group SU(2), at $\beta = 1.75$, $\kappa = 0.165$ and $c_{\rm sw} = 0$. We compare $\Omega_l(\hat{Q})$ computed in 32-bit precision (solid line) with the one computed in 64-bit precision (dashed line). We remark at this point that, although matrix multiplications are performed in 32- or 64-precision, scalar products are always evaluated with 64-bit precision. As starting vector Φ_{in} we use a Gaussian random vector.

The picture confirms our expectations: as long as the values of $\Omega_l(\hat{Q})$ are growing with l, both curves are basically identical. When $\Omega_l(\hat{Q})$ starts to decrease, however, the values for $\Omega_l(\hat{Q})$ obtained with 32-bit precision deviate strongly from the ones obtained with 64-bit precision. In fact, instead of decreasing, $\Omega_l(\hat{Q})$ computed with the 32-bit precision version of the program, even increases and

eventually runs into a numerical overflow. This is no surprise, of course: as discussed above, when $\Omega_l(\hat{Q}) \ll \Omega_{l-1}(\hat{Q})$, large cancellations must occur, leading to the observed precision losses. Note, moreover, that even the values for $\Omega_l(\hat{Q})$ obtained with a 64-bit precision are affected by large rounding errors: the final value $\Omega_{2n+1}(\hat{Q})$ is completely wrong, namely $O(10^{25})$ instead of $O(1)$, as it should be. Clearly, using only 64-bit precision reduces the precision losses (no overflow is observed), but it certainly is not sufficient to keep in general rounding errors on a tolerable level.
Figure 3 shows how the different, improved ordering schemes help. In Fig.3a we plot $\omega_l(q)$ for three ordering schemes, the subpolynomial (solid line), the bit-reversal (dashed line) and Montvay's scheme (dash-dotted line). We only show the curves for $q = 1$, i.e. the worst case in Fig.1a. For other values of q the picture looks very similar. As compared with Fig. 1a, the large oscillations are strongly suppressed. As a consequence, when now $\Omega_l(\hat{Q})$ is constructed with 32-bit precision, according to the improved ordering schemes, numerical overflows are avoided and the desired fit accuracy is reached, as demonstrated in Fig.3b.

4 Quantitative tests

After the qualitative tests of the ordering schemes discussed in the previous section, we would now like to turn to more quantitative results. Guided by the observation that the evaluation of a polynomial in a single variable gives information on the precision losses that may occur when evaluating the same polynomial in a matrix, we will first investigate single variable estimators for rounding errors. Then we will discuss the rounding-error effects that arise in the numerical construction of the polynomial $\hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2)$. The results of this section only refer to ordering schemes for polynomials in the product representation of eq.(10). We stress that there would be no qualitative difference in our results if the representation eq.(13) had been taken.

4.1 Estimators of rounding errors for a single variable

A possible way of defining single variable estimators for rounding-error effects consists in quantifying the magnitude of the oscillations of $\omega_l(q)$. As a first step, let us evaluate for a given l the maximal and the minimal value of $|P^l_{n,\epsilon}(s)|$, eq.(12), over the interval $0 \leq s \leq 1$. The ratio of the maximal to the minimal value, i.e. $\tilde{R}_l = \max_{s\in[0,1]} |P^l_{n,\epsilon}(s)| / \min_{s\in[0,1]} |P^l_{n,\epsilon}(s)|$, is then a measure of how different the order of magnitude of the l-th partial product can be for different values of s. Building the maximum of $\tilde{R}_l$ with respect to l, we get a quantity independent of l and s:

$$R_{\max} = \max_{l\in\{1,\ldots,n\}} \left\{ \frac{\max_{s\in[0,1]} |P^l_{n,\epsilon}(s)|}{\min_{s\in[0,1]} |P^l_{n,\epsilon}(s)|} \right\} \ . \tag{24}$$

It is clear that $R_{\max}$ has to be smaller than the largest representable number on a given computer to guarantee the stability of the evaluation of the full polynomial. This is actually sufficient to exclude the occurrence of overflows or underflows in the evaluation of the considered partial products. If $R_{\max}$ fulfils this condition but still assumes large values, in the case of a polynomial in a matrix a reliable numerical result cannot be expected, since rounding errors are likely to lead to a substantial loss of precision. Moreover, even if $R_{\max}$ is reasonably small, it is still possible that the quantity $\max_{s\in[0,1]} |P^l_{n,\epsilon}(s)|$ shows large oscillations as a function of l. As a consequence, another quantity of interest is the maximum value of the partial products itself:

$$M_{\max} = \max_{s\in[0,1], l\in\{1,\ldots,n\}} |P^l_{n,\epsilon}(s)| \ . \tag{25}$$

This again has to be smaller than the largest representable number in order not to run into overflow. Note that $R_{\max}$ and $M_{\max}$ are computed for $s \in [0,1]$, whereas the polynomial in eq.(10) has a given relative fit accuracy only in the interval $s \in [\epsilon, 1]$. However, as will be explicitly demonstrated below, our results for $R_{\max}$ and $M_{\max}$ do not depend very much on the choice of the lower end of the interval. A final remark is that the values of $R_{\max}$ and $M_{\max}$ should be taken only as a first indication for the size of the rounding errors. Since it is not guaranteed that in a practical case, for the polynomial in a matrix, the value of $R_{\max}$ or $M_{\max}$ is actually assumed, it is very possible that $R_{\max}$ and $M_{\max}$ may overestimate the rounding errors. On the other hand, since in the case of a polynomial in a matrix the rounding errors occurring in different intermediate steps of the numerical computation can easily accumulate, $R_{\max}$ and $M_{\max}$ might also yield an underestimate.

In order to compute the values of $R_{\max}$ and $M_{\max}$ we take 5000 values of s, equally spaced in the interval $[0,1]$. We explicitly checked that the values of $R_{\max}$ do not depend very much on the lower end of the interval $[s_{\min},1]$ from which s is taken. In Fig. 4 we show $R_{\max}$, in the case of the bit-reversal scheme, as a function of the lower end of the interval $s_{\min}$, measured in units of the parameter ϵ. The data refer to polynomials of different degree, $n=30$, $n=86$ and $n=146$, with the parameter ϵ determined by the fixed relative fit accuracy $\delta=0.001$. As Fig. 4 shows, the dependence of $R_{\max}$ on $s_{\min}$ is very weak. For the other ordering schemes, we find a similar behaviour of $R_{\max}$ as a function of $s_{\min}$. This justifies the use of $s_{\min}=0$ that we have adopted for the numerical tests described below.
We start by comparing the subpolynomial and the bit-reversal schemes, as they are closely related to each other. In Fig. 5 we show $R_{\max}$ and $M_{\max}$ as a function of the degree n of the polynomial $P_{n,\epsilon}(s)$ of eq.(10), keeping the relative fit accuracy $\delta=0.1$ constant by adjusting the parameter ϵ. For the subpolynomial scheme, the divisor m is chosen to be $m\approx\sqrt{n}$. Figure 5 clearly confirms our expectation that the bit-reversal scheme, considered as a generalization of the subpolynomial ordering scheme, gives smaller values of $R_{\max}$ and $M_{\max}$. For degrees of the polynomial $n>40$ the "dangerous" oscillations are substantially suppressed in the bit-reversal scheme compared with the subpolynomial scheme.
In Fig. 6 we show the values of $R_{\max}$ and $M_{\max}$ for the bit reversal, the naive, the pairing and the Montvay's scheme, at a fixed value of $\delta=0.001$, as a function of n. Numerical tests for the different ordering schemes were also performed at values of $\delta=0.1$ and $\delta=0.01$, yielding a very similar qualitative behaviour of $R_{\max}$ and $M_{\max}$ as a function of n.
The first striking observation in Fig. 6 is that one obtains with the naive ordering, already for moderate degrees $n\approx 30$ of the polynomial, large values of $R_{\max}$ and $M_{\max}$; this indicates that very large oscillations of the partial products occur in the intermediate steps of the construction of the polynomial. Using the naive scheme on machines with 32-bit precision or even with 64-bit precision, a safe evaluation of the polynomial in the product representation can certainly not be guaranteed.
The behaviour of the values of $R_{\max}$ and $M_{\max}$ obtained by using the naive ordering scheme clearly demonstrates the necessity of finding better ordering schemes. That such ordering schemes do exist is also demonstrated in Fig. 6. For $n<100$,

the values for $R_{\max}$ and $M_{\max}$ obtained from the pairing, bit reversal and Montvay's schemes are close to each other and many orders of magnitude below the ones of the naive scheme. However, for $n > 120$, the values of $R_{\max}$ from these ordering schemes also start to deviate from each other. Taking the values of $R_{\max}$ and $M_{\max}$ as an estimate of the effects of rounding errors arising in the construction of polynomials in matrix, it seems that the bit reversal and Montvay's scheme are the most effective, in reducing these effects, out of the ordering schemes investigated here.

4.2 Quantitative tests for a polynomial in a matrix

The numerical tests involving a polynomial in the matrix $\hat{Q}$, eq.(35), are performed using a sample of thermalized SU(3) gauge field configurations on $8^3 \cdot 16$ lattices. All numerical computations were done on the massively parallel Alenia Quadrics (APE) machines, which have only 32-bit precision. Simulation parameters are chosen to be $\beta = 6.8$, $\kappa = 0.1343$ and $c_{\text{sw}} = 1.42511$. They correspond to realistic parameter values as actually used in simulations to determine values of c_{sw} non-perturbatively [21]. Throughout this section we will use Schrödinger functional boundary conditions as mentioned in Appendix A. Averaging over the gauge field configurations, for the above choice of parameters and $c_M = 0.735$, the lowest eigenvalue of $\hat{Q}^2$ is $\lambda_{\text{min}} = 0.00114(4)$ and the largest is $\lambda_{\text{max}} = 0.8721(3)$. Investigations are performed at values of (n, ϵ) of $(16, 0.003)$, $(32, 0.003)$, $(64, 0.0022)$ and $(100, 0.0022)$. At each of these values of (n, ϵ) we have generated O(50) gauge field configurations. Governed by heuristic arguments [15], for a real simulation, ϵ should be chosen as $\epsilon \approx 2\lambda_{\text{min}}$ and $\delta \approx 0.01$, which roughly corresponds to the choice $(n, \epsilon) = (64, 0.0022)$.
We apply the matrix $\hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2)$, which should be close to the unit matrix for our choices of n and ϵ, to a random Gaussian vector R_G and construct the vectors

$$\Phi_{\text{order}} = \hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2) R_G \ , \tag{26}$$

where $P_{n,\epsilon}(\hat{Q}^2)$ is defined analogously to eq.(10) and evaluated using different ordering schemes. The subscript "order" stands for naive, pairing, bit reversal, Montvay and Clenshaw. In the latter case, the polynomial $\hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2)$ is, of course, constructed by using the Clenshaw recursion relation, as explained in section 3.2. Following this prescription, one first constructs the Chebyshev polynomial T^*_{n+1}, eq.(5), with s replaced in the obvious way by the matrix $\hat{Q}^2$. Since the polynomial we are finally interested in is given by $\hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2) = 1 + \rho T^*_{n+1}$, see eq.(6), any rounding error that is induced in the construction of T^*_{n+1} is suppressed by a factor of $\mathrm{O}(\delta)$ since $|\rho| \leq \delta$.

On a given gauge field configuration and for a given R_G we compute

$$\Delta = \frac{1}{\sqrt{N}} \|\Phi_{\mathrm{order}} - \Phi_{\mathrm{Clenshaw}}\| \ , \tag{27}$$

where N is the number of degrees of freedom of the vector Φ and $\|.\|$ denotes the square root vector norm.

Since the Clenshaw recurrence is believed to be the numerically most stable method to evaluate linear combinations of Chebyshev polynomials, the values of Δ can be interpreted as a measure for the effects of rounding errors in the evaluation of Φ_{order}. The results for Δ as a function of n are shown in Fig. 7. Using the naive ordering scheme, we could not run the cases of $n = 64$ and $n = 100$ because of numerical overflows. When adopting the pairing scheme, Δ takes large values for $n = 64$ and $n = 100$. The bit-reversal scheme gives small but non-negligible values of Δ for the considered values of n. Finally, Montvay's scheme yields Δ of order 10^{-6} for all values of n. We conclude that, somewhat surprisingly, it is possible to find ordering schemes through which the construction of the polynomial $\hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2)$ can be done with a precision that is comparable with the one expected when performing $O(n)$ multiplications in 32-bit arithmetics. As already observed, one may evaluate the corresponding quantity Δ also for the product representation eq.(13) [16]. In this case again, small values of $\Delta \approx O(10^{-6})$ are found when using, however, *either* Montvay's or bit-reversal ordering schemes. This indicates that the improvement achieved through these ordering schemes may also depend to some extent on the chosen product representation.

5 32-bit versus 64-bit precision

In order to obtain Δ in eq.(27), the vector Φ_{Clenshaw} has been computed using solely 32-bit precision. It might be asked therefore, whether Δ can really be considered as a measure of the size of the rounding errors occurring in the construction of Φ_{order}. To the best of our knowledge, the theorem stated in [17], providing an upper bound on the rounding errors that occur in the use of the Clenshaw recurrence, is not straightforwardly generalizable to the case of a polynomial in a matrix. Hence it can not be guaranteed that, in the case of a polynomial in a matrix, the Clenshaw recursion is providing us with a reference method to quantify the rounding errors on $\Phi_{\rm order}$. In order to clarify this point, we decided to compare results obtained from 32-bit precision with those obtained from 64-bit precision programs. For this test we considered the SU(2) gauge group with the same bare parameters as specified in section 3.3, using three different lattice sizes. All the results reported in this section were obtained on a single thermalized gauge configuration and hence are given without a statistical error. However, we checked explicitly in some cases that using different gauge configurations yields only negligible changes in the results discussed here.

In the following, we denote by

$$\chi_{\rm order} \equiv \hat{Q}\sqrt{c_{2n}}(\hat{Q}-r_{2n})\cdot\ldots\cdot\sqrt{c_1}(\hat{Q}-r_1)\hat{Q}R_G \tag{28}$$

the vector obtained with 32-bit precision and by $\tilde{\chi}_{\rm order}$ the corresponding vector obtained with 64-bit precision. The subscript "order" labels again different ordering schemes and R_G is a random vector obtained from a Gaussian distribution with unit variance. Analogously, we denote by $\chi_{\rm Clenshaw}$ and $\tilde{\chi}_{\rm Clenshaw}$ the vectors obtained using the Clenshaw recursion with the two precisions.

Then the norm of the difference between the vectors $\chi_{\rm order}$ and $\tilde{\chi}_{\rm order}$,

$$\eta_{\rm order} = \frac{1}{\sqrt{N}}\|\chi_{\rm order} - \tilde{\chi}_{\rm order}\| \; , \tag{29}$$

with N the number of degrees of freedom of the vector χ, can serve as an estimate of the rounding errors when $\chi_{\rm order}$ is evaluated. Let us start by comparing the values of $\eta_{\rm order}$ for the subpolynomial (sp), bit reversal (br) and Montvay's ordering schemes, taking $n = 64$ and $\epsilon = 0.0015$:

$$\eta_{\rm sp} \simeq 3.7\cdot 10^{-5} \quad , \quad \eta_{\rm br} \simeq 4.3\cdot 10^{-6} \quad , \quad \eta_{\rm Montvay} \simeq 5.5\cdot 10^{-6} \quad . \tag{30}$$

The above values, in particular for the bit reversal and Montvay's ordering scheme, are close to the precision level that is expected to be reached optimally on 32-bit machines after performing $2n+2 = 130$ multiplications.

Of particular interest are the values of $\eta_{\rm Clenshaw}$. Since we expect that the vector χ is most precisely evaluated when the Clenshaw recurrence relation is used with

64-bit precision, the values of η_{Clenshaw} would tell us how much the computation of χ_{Clenshaw} is affected by rounding errors. The results are shown in table 1. We compare also χ_{order} with $\tilde{\chi}_{\text{Clenshaw}}$, using the bit-reversal scheme as an example for an ordering scheme and evaluate

$$\hat{\eta}_{\text{br}} = \frac{1}{\sqrt{N}} \|\chi_{\text{br}} - \tilde{\chi}_{\text{Clenshaw}}\| \,, \tag{31}$$

again reporting the results in table 1.

Table 1: The quantities η_{Clenshaw}, eq.(29), and $\hat{\eta}_{\text{br}}$, eq.(31), for various parameters of the polynomial $P_{n,\epsilon}$. The values in the table are computed using one thermalized SU(2) gauge configuration at $\beta = 1.75$, $\kappa = 0.165$ and $c_{\text{sw}} = 0$. Different lattices sizes $L^3 \cdot T$ are compared.

$L^3 \cdot T$	n	ϵ	δ	η_{Clenshaw}	$\hat{\eta}_{br}$
$8^3 \cdot 16$	16	0.0215	0.013	$8.4 \cdot 10^{-8}$	$1.6 \cdot 10^{-6}$
$8^3 \cdot 16$	32	0.0058	0.013	$1.3 \cdot 10^{-7}$	$2.1 \cdot 10^{-6}$
$8^3 \cdot 16$	64	0.0015	0.013	$2.7 \cdot 10^{-7}$	$4.3 \cdot 10^{-6}$
$8^3 \cdot 16$	100	0.0006	0.014	$6.0 \cdot 10^{-7}$	$9.3 \cdot 10^{-6}$
$4^3 \cdot 8$	64	0.0015	0.013	$2.7 \cdot 10^{-7}$	$4.6 \cdot 10^{-6}$
$6^3 \cdot 16$	64	0.0015	0.013	$2.7 \cdot 10^{-7}$	$4.5 \cdot 10^{-6}$
$8^3 \cdot 16$	64	0.0005	0.109	$2.1 \cdot 10^{-6}$	$5.6 \cdot 10^{-6}$
$8^3 \cdot 16$	64	0.0001	0.545	$9.8 \cdot 10^{-6}$	$8.7 \cdot 10^{-6}$

For practical values of $\delta < 0.014$ the Clenshaw recursion relation turns out to be at least one order of magnitude more precise than the bit-reversal scheme. In fact, the magnitude of the rounding error using the Clenshaw recursion looks in some cases even better than what is naively expected from using 32-bit precision. This effect is easily explained by the fact that the rounding error is suppressed by a factor of O(δ), as discussed in the previous section. In addition, one can observe that, as δ increases, the magnitude of the rounding error, as measured by η_{Clenshaw}, also grows and eventually reaches values of the same order as for the bit-reversal scheme when δ becomes of $O(1)$. We conclude that for $\delta < 0.1$ the Clenshaw recursion relation allows for a very precise evaluation of $\hat{Q}^2 P_{n,\epsilon}(\hat{Q}^2)$, even when only 32-bit arithmetic is employed. This allows to test the different ordering schemes and to obtain reliable estimates of the rounding errors associated to these schemes by using solely 32-bit precision.

Table 1 also shows that the magnitude of rounding errors, as estimated by $\hat{\eta}_{br}$ for the bit-reversal scheme, increases moderately with the degree n of the polynomial (with ϵ tuned to keep δ constant), but do not significantly depend on the considered lattice sizes. The problem of the lattice size dependence of rounding errors has not been systematically investigated. However, on lattices not larger than $8^3 \cdot 16$, we could not observe any significant dependence of the rounding errors, measured through η and $\hat{\eta}$, on the lattice size. This might be explained by the fact that $\hat{Q}$ has a band structure with only the diagonal and some off-diagonal matrix elements not vanishing. For this argument to be valid, however, the precision losses, characterized by η, occurring in a single multiplication by $\hat{Q}$, should be small enough for the propagation of their effect through subsequent multiplications to be negligible. In practice we have found that when η reaches a level of $\eta \approx O(10^{-4})$, a significant growth of the size of rounding errors, estimated by η itself, can be observed as the lattice volume increases. When such a behaviour sets in, it is certainly advisable to switch from 32-bit to 64-bit arithmetics.

6 Conclusions

In a class of fermion simulation algorithms, relying on the multiboson technique, polynomials in a matrix have to be numerically evaluated. In a number of cases the polynomials are needed in their product representation, in order to achieve an efficient performance of the algorithms. Moreover, the simulations are often done on machines having only 32-bit precision. However, it is well known that, in evaluating a polynomial using its product representation, great care has to be taken, since rounding errors can easily lead to significant precision losses and even numerical instabilities.

In this paper we investigated the effects of various ordering schemes for the monomial factors or, equivalently, the complex roots in the numerical construction of a polynomial according to a product representation. We found that different ordering schemes can lead to rounding-error effects ranging from numerical overflow to retaining a precision comparable to the one that can be provided by the use of numerically stable recurrence relations.

In the case of a polynomial of a single, real variable s, approximating the function $1/s$, we introduced estimators for the rounding errors, which give qualitative information about the level of precision losses occurring when a polynomial in a matrix is considered. These estimators indicate that the bit-reversal scheme and a scheme suggested by Montvay can keep rounding-error effects to a low level, for polynomials in a matrix of order up to $n \approx 200$.

Considering cases relevant for numerical simulations of lattice QCD, we studied the magnitude of the precision losses affecting the evaluation of polynomials in a particular matrix, when adopting different ordering schemes for the monomials of the product representation. We found that Montvay's ordering scheme seems to be particularly suited for this problem: by adopting this scheme, the rounding errors could be kept at a level that is comparable to the one that is reached when using the stable Clenshaw recurrence relation.

As the most important outcome of our investigation, we conclude that there exist orderings of the roots that allow a numerically precise evaluation of a polynomial in a matrix, even up to degrees n of the polynomial of O(100). Although in this work only a particular matrix, relevant for simulations of lattice QCD, has been considered, we think that the main features of our results should hold true for several different kinds of matrices and that, also, product representations can be used in more general situations while keeping rounding-error effects at a tolerable level.

Acknowledgements

This work is part of the ALPHA collaboration research programme. We are most grateful to S. Sint and U. Wolff for a critical reading of the manuscript. We thank DESY for allocating computer time to this project. R.F. thanks the Alexander von Humboldt Foundation for the financial support to his research stay at DESY–Hamburg, where part of this work was done.

A Definition of the matrix $\hat{Q}$

An application, where a product representation of polynomials in a matrix is used, is the Monte Carlo simulation of fermion systems by the multiboson technique and related algorithms. For completeness, we will give in this appendix an explicit definition of the matrix $\hat{Q}$ used for the tests presented in this paper.

Let us consider a Euclidean space-time lattice with lattice spacing a and size $L^3{\cdot}T$. With the lattice spacing set to unity from now on, the points on the lattice have integer coordinates (t, x_1, x_2, x_3). A gauge field $U_\mu(x) \in SU(N_c)$ is assigned to the link pointing from the site x to the site $(x + \mu)$, where $\mu = 0, 1, 2, 3$ denotes the four forward directions in space-time. In this paper we will use $N_c = 2, 3$. For the case of $N_c = 2$, the SU(2) gauge group, the boundary conditions are chosen to be periodic in all directions. For $N_c = 3$, the SU(3) gauge group, we adopt periodic boundary conditions in space and Schrödinger functional boundary conditions in the time direction [22]. After integrating out the fermion fields, their effects appear in the resulting effective theory through the determinant of a matrix $A = A[U]$, depending on the gauge field configuration U. In the case of the $SU(N_c)$ gauge theory with $n_f = 2$ mass degenerate quark species, which is considered here, we have $A = Q^2$, where the matrix Q, characterizing the fermion lattice action, is taken as follows:

$$\begin{aligned} Q[U]_{xy} = {} & \frac{c_0}{c_M}\gamma_5\Big[(1 + \sum_{\mu\nu}[\frac{i}{2}c_{\text{sw}}\kappa\sigma_{\mu\nu}\mathcal{F}_{\mu\nu}(x)])\delta_{x,y} \\ - {} & \kappa\sum_{\mu}\{(1-\gamma_\mu)U_\mu(x)\delta_{x+\mu,y} + (1+\gamma_\mu)U_\mu^\dagger(x-\mu)\delta_{x-\mu,y}\}\Big] \ . \end{aligned} \tag{32}$$

Here κ and c_{sw} are parameters that have to be chosen according to the physical problem under consideration, $c_0 = (1 + 8\kappa)^{-1}$, and c_M is a constant serving to optimize simulation algorithms. Typically $\kappa \approx 1/8$ and both c_{sw} and c_M are O(1). In order to speed up the Monte Carlo simulation, it is not the original matrix Q but an even-odd preconditioned matrix $\hat{Q}$ that is used. Let us rewrite the matrix Q in eq. (32) as

$$Q \equiv \frac{c_0}{c_M}\gamma_5 \begin{pmatrix} 1 + T_{ee} & M_{eo} \\ M_{oe} & 1 + T_{oo} \end{pmatrix} \ , \tag{33}$$

where we introduce the matrix $T_{ee}(T_{oo})$ acting on vectors defined on the even (odd) sites:

$$(T)_{xa\alpha,yb\beta} = \sum_{\mu\nu}[\frac{i}{2}c_{sw}\kappa\sigma^{\alpha\beta}_{\mu\nu}\mathcal{F}^{ab}_{\mu\nu}(x)\delta_{xy}] \ . \tag{34}$$

The off-diagonal parts M_{eo} and M_{oe} connect the even with odd and odd with even lattice sites, respectively. Preconditioning is now realized by writing the determinant of Q, apart from an irrelevant constant factor, as

$$\begin{aligned} \det(Q) &\propto \det(1+T_{ee})\det\hat{Q} \\ \hat{Q} &= \frac{\hat{c}_0}{c_M}\gamma_5(1+T_{oo}-M_{oe}(1+T_{ee})^{-1}M_{eo}) \ . \end{aligned} \tag{35}$$

The constant factor $\hat{c}_0$ is given by $\hat{c}_0 = 1/(1+64\kappa^2)$, and the constant c_M is chosen such that the eigenvalues of $\hat{Q}$ are in the interval $[-1,1]$.

The matrices $\sigma_{\mu\nu}, \ \mu,\nu = 0,...,3$ in the above expressions, are defined by the commutator of γ-matrices

$$\sigma_{\mu\nu} = \frac{i}{2}\left[\gamma_\mu,\gamma_\nu\right] \ . \tag{36}$$

The $4\otimes 4$ γ-matrices are given by

$$\gamma_\mu = \begin{pmatrix} 0 & e_\mu \\ e_\mu^\dagger & 0 \end{pmatrix} \ , \tag{37}$$

with the $2\otimes 2$ matrices

$$e_0 = -1 \ \ e_j = i\sigma_j, \ j = 1,2,3 \tag{38}$$

and the Pauli-matrices σ_j

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \ \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \ \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \ . \tag{39}$$

The matrix $\gamma_5 = \gamma_0\gamma_1\gamma_2\gamma_3$ is thus diagonal

$$\gamma_5 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \ . \tag{40}$$

$\mathcal{F}_{\mu\nu}(x)$ is a tensor antisymmetric under the exchange $\mu \leftrightarrow \nu$ and, for given values of μ,ν, an anti-Hermitian $N_c \otimes N_c$ matrix depending on the gauge links in the vicinity of the lattice site x:

$$
\begin{aligned}
\mathcal{F}_{\mu\nu}(x) &= \frac{1}{8}\Big[U_\mu(x)U_\nu(x+\hat\mu)U^\dagger_\mu(x+\hat\nu)U^\dagger_\nu(x) \\
&+ U_\nu(x)U^\dagger_\mu(x+\hat\nu-\hat\mu)U^\dagger_\nu(x-\hat\mu)U_\mu(x-\hat\mu) \\
&+ U^\dagger_\mu(x-\hat\mu)U^\dagger_\nu(x-\hat\nu-\hat\mu)U_\mu(x-\hat\nu-\hat\mu)U_\nu(x-\hat\nu) \\
&+ U^\dagger_\nu(x-\hat\nu)U_\mu(x-\hat\nu)U_\nu(x-\hat\nu+\hat\mu)U^\dagger_\mu(x) \\
&- \text{h.c.}\Big] \ . \qquad (41)
\end{aligned}
$$

References

[1] M. Lüscher, Nucl. Phys. B418 (1994) 637.

[2] B. Jegerlehner, Nucl. Phys. B465 (1996) 487.

[3] P. de Forcrand, Nucl. Phys. B (Proc. Suppl.) 47 (1996) 228.

[4] A. Borrelli, P. de Forcrand and A. Galli, Nucl. Phys. B477 (1996) 809.

[5] K. Jansen, B. Jegerlehner and C. Liu, Phys. Lett. B375 (1996) 255.

[6] K. Jansen, Nucl. Phys. B (Proc. Suppl.) 53 (1997) 127.

[7] U. Wolff, Nucl. Phys. B (Proc.Suppl.) 63 (1998) 937, hep-lat/9708018.

[8] S. Elser and B. Bunk, Nucl. Phys. B (Proc.Suppl.) 63 (1998) 940, hep-lat/9709121.

[9] S. Duane, A. D. Kennedy, B. J. Pendleton and D. Roweth, Phys. Lett. B195 (1987) 216.

[10] A. M. Horowitz, Phys. Lett. B156 (1985) 89; Nucl. Phys. B280 (1987) 510; Phys. Lett. 268B (1991) 247.

[11] K. Jansen and C. Liu, Nucl. Phys. B453 (1995) 375.

[12] I. Montvay, Nucl. Phys. B (Proc.Suppl.) 63 (1998) 10, hep-lat/9709080.

[13] P. Sawicki, Kraków preprint, TPJU-1/97, hep-lat/9703005.

[14] P. de Forcrand and T. Takaishi, Nucl. Phys. B (Proc. Suppl.) 53 (1997) 968.

[15] R. Frezzotti and K. Jansen, Phys. Lett. B402 (1997) 328; Nucl. Phys. B (Proc.Suppl.) 63 (1998) 943, hep-lat/9709033.

[16] R. Frezzotti and K. Jansen, paper in preparation.

[17] L. Fox and I. B. Parker, *Chebyshev polynomials in numerical analysis*, Oxford University Press, London, 1968.

[18] B. Bunk, K. Jansen, B. Jegerlehner, M. Lüscher, H. Simma and R. Sommer, Nucl. Phys. B (Proc. Suppl.) 42 (1995) 49, hep-lat/9411016.

[19] I.Montvay, DESY preprint, DESY 97-132, hep-lat/9707005.

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes*, Second Edition, Cambridge University Press, Cambridge, 1992.

[21] K. Jansen and R. Sommer, Nucl. Phys. B (Proc.Suppl.) 63 (1998) 85, hep-lat/9709022, CERN preprint, CERN-TH/98-84, hep-lat/9803017.

[22] M. Lüscher, S. Sint, R. Sommer, P. Weisz and U. Wolff, Nucl. Phys. B491 (1997) 232.

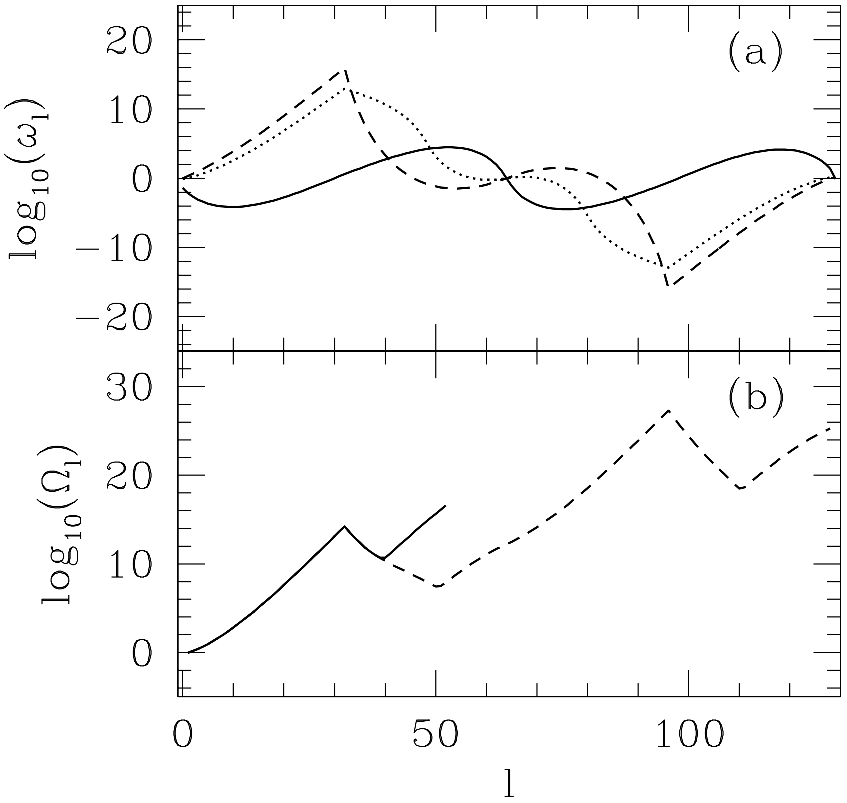


Figure 1: The behaviour of $\omega_l(q)$ in eq.(15) as a function of l (a). The three curves correspond to $q^2 = \epsilon$ (full line), $q^2 = (1+\epsilon)/2$ (dotted line) and $q^2 = 1$ (dashed line). In (b) we give $\Omega_l(\hat{Q})$ of eq.(15) as a function of l, by computing $\Omega_l(\hat{Q})$ once with 32-bit (full line) and once with 64-bit (dashed line) arithmetics.

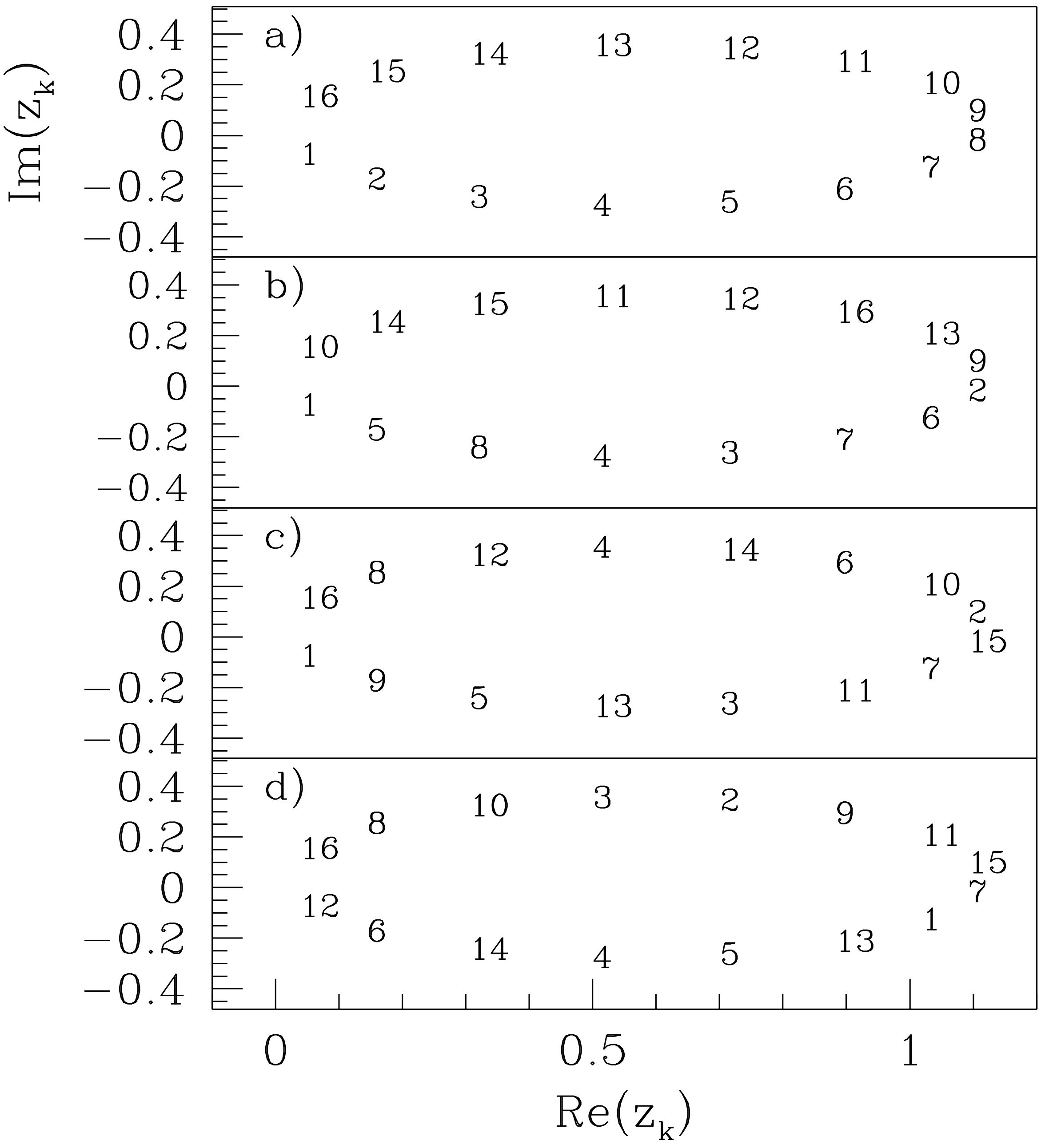


Figure 2: The roots z_k, eq. (9), with $k = 1, \ldots, 16$ and $\epsilon = 0.1$ are shown in the complex plane. Labels of roots indicate in which order they are used for the numerical evaluation of the polynomial, eq. (10), within each ordering scheme. We show in a) the naive, b) the pairing, c) the bit reversal and d) the Montvay's ordering scheme.

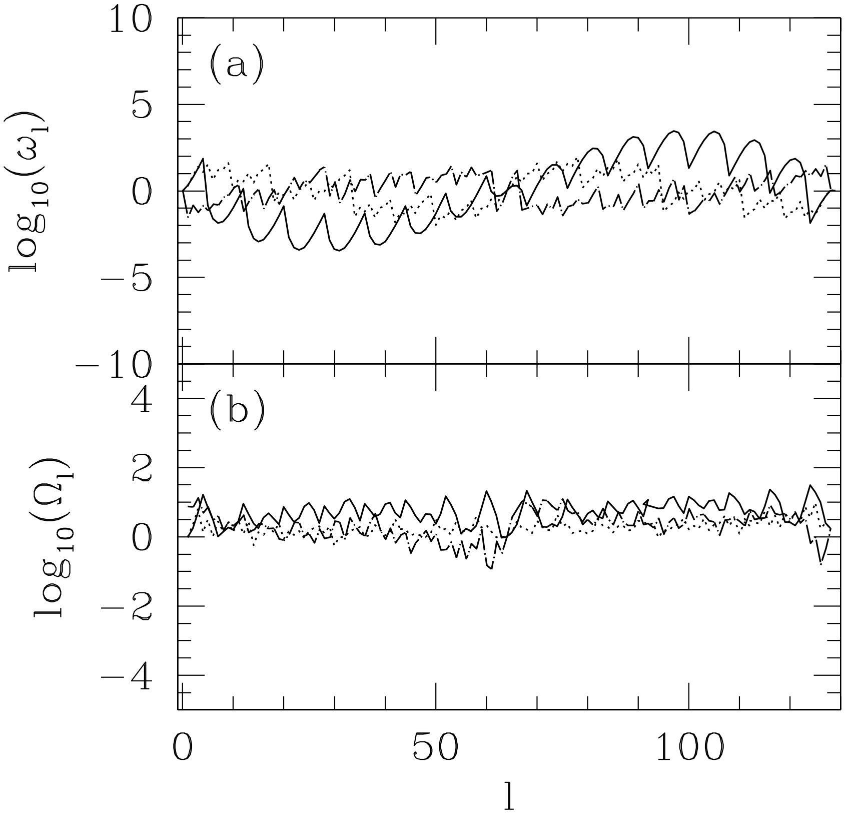


Figure 3: The behaviour of (a) $\omega_l(q)$, eq.(15), and (b) of $\Omega_l(\hat{Q})$, eq.(15), as a function of l. We use the subpolynomial (solid line), the bit reversal (dotted line) and Montvay's (dash-dotted line) ordering schemes. For Fig.3a, we have chosen only $q = 1$. All results shown in the figure are obtained using 32-bit precision.

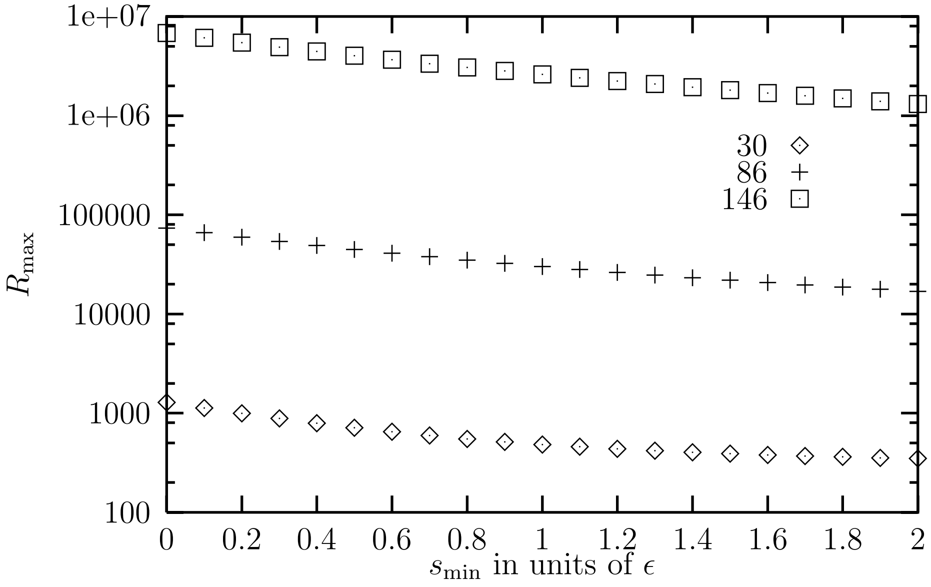


Figure 4: R_{max}, eq. (24), is shown as a function of s_{min}/ϵ, where s_{min} is the lower end of the interval $[s_{\text{min}}, 1]$ from which the values of s are taken to compute R_{max}. We show data for three polynomials of different degree, $n = 30$, $n = 86$ and $n = 146$ at fixed relative fit accuracy $\delta = 0.001$, using the bit-reversal scheme. Although different in magnitude, R_{max} shows also for the other schemes a very similar flat behaviour as a function of s_{min}/ϵ.

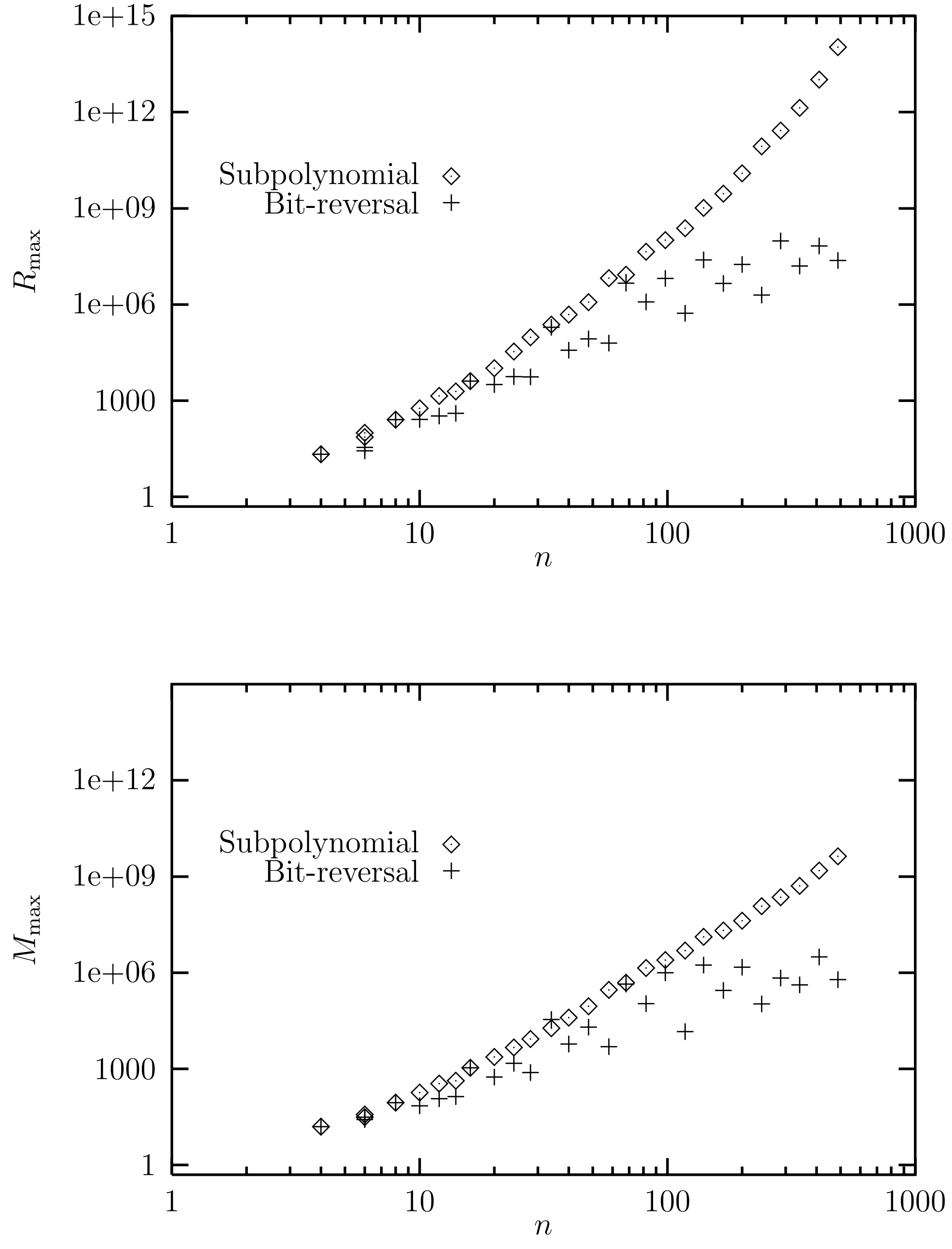
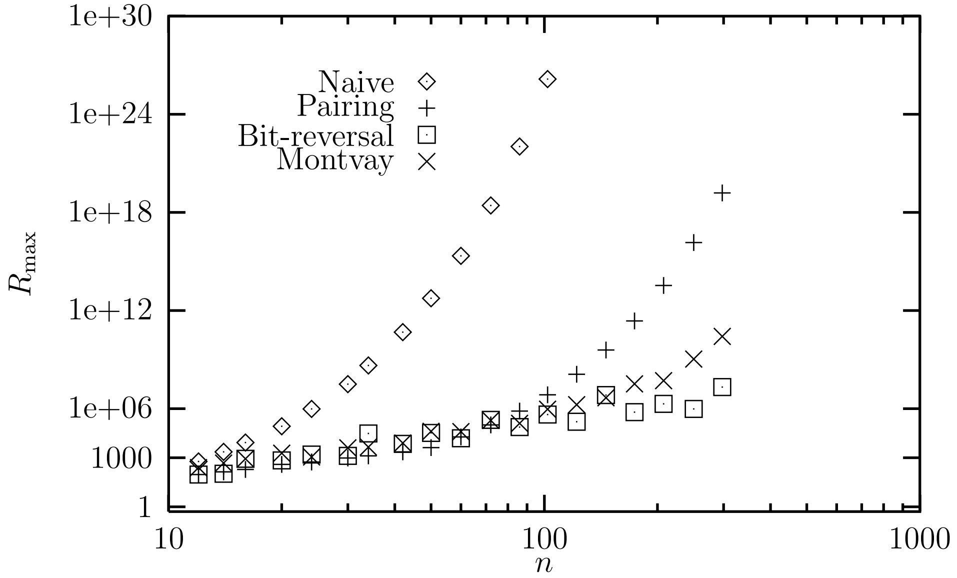


Figure 5: $R_{\max}$, eq. (24), and $M_{\max}$, eq. (25), are shown as a function of the degree of the polynomial at fixed relative fit accuracy $\delta = 0.1$. We compare subpolynomial and bit-reversal ordering schemes.



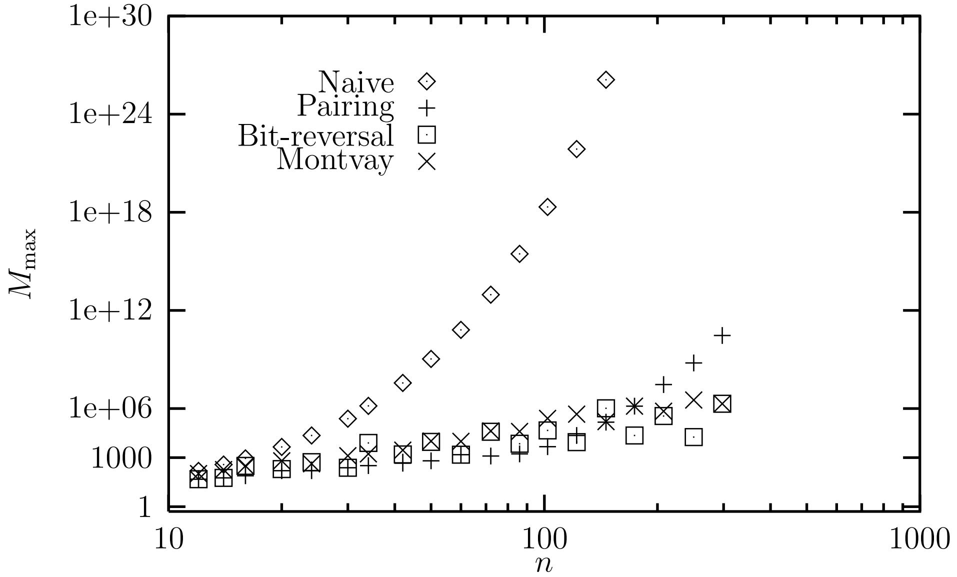


Figure 6: $R_{\rm max}$, eq. (24), and $M_{\rm max}$, eq. (25), are shown as a function of the degree of the polynomial at fixed relative fit accuracy $\delta = 0.001$. We compare naive, pairing, bit reversal and Montvay's ordering schemes.

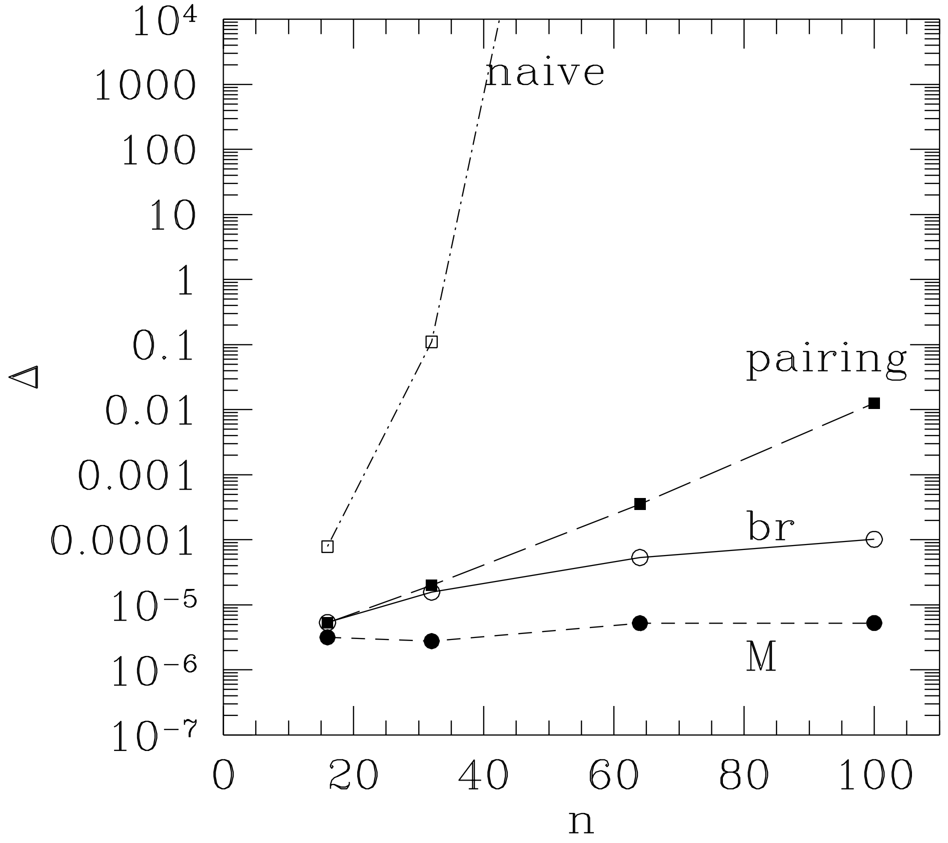


Figure 7: The quantity Δ, eq. (27), is shown as a function of the degree n of the polynomial occurring in its definition. We compare the naive, pairing, bit reversal (br) and Montvay's (M) ordering schemes.